Fast and Accurate Generalized Harmonic Analysis Using Newton's Method

Hisayori Noda* and Akinori Nishihara†
Tokyo Institute of Technology, 2-12-1-W9-108 Ookayama, Meguro-ku, Tokyo, 152-8552 Japan
* E-mail: hisayori@nh.cradle.titech.ac.jp Tel: +81-3-5734-3232
† E-mail: aki@cradle.titech.ac.jp Tel: +81-3-5734-3232

***Abstract*—A fast and accurate method for Generalized Harmonic Analysis (GHA) is proposed. The proposed method estimates the parameters of a sinusoid and subtracts it from a target signal one by one. The frequency of the sinusoid is estimated around a peak of Fourier spectrum using Newton's method, which is much faster than search methods. The amplitude and the phase are estimated to minimize the squared sum of the residue after extraction of estimated sinusoids from the target signal.**

A method to improve the accuracy of sinudoidal parameters using multi-dimensional Newton's method is also proposed. This method is applied to the extracted sinusoidal parameters and minimizes the partial derivative vector of the squared error.

Audio signals are analyzed by the proposed methods, which confirms the accuracy compared to the previous method.

I. Introduction

Generalized harmonic analysis (GHA) [1] is a concept of signal analysis introduced by N. Wiener in 1930, in which a target signal is expressed as sum of sinusoids. Each sinusoid has three parameters; frequency, amplitude, and phase. Unlike short-time Fourier Transform, the frequency is not restricted to multiples of inverse of the frame size. So the frequency resolution is very high by its nature, and the time resolution is also high because the frame size can be made short without affecting the frequency resolution.

A target signal is divided into frames having size N. The signal in a frame is approximated by the sum of sinusoids as

$$x_0(n) \quad \cong \quad \sum_{k=1}^{K} A_k \sin(\omega_k n + \phi_k), \qquad (1)$$

where A_k, ω_k and ϕ_k are the amplitude, the angular frequency and the phase of the k-th sinusoid, respectively, and K is the number of sinusoids to be extracted. We can easily synthesize the signal using the sinusoidal parameters. We estimate these parameters so as to minimize the difference between the target signal and the synthesized signal.

Actually GHA is an approach of the sinusoidal parameters estimation and it is related to the frequency estimation. In the research field of the frequency estimation, there are many well-developed methods [14].

Periodgram [16] is a basic frequency estimation technique. It finds the peak of FFT spectrum and estimates it as the frequency. The frequency resolution is not high because the estimated frequency is restricted to multiples of inverse of the frame size.

The Pisarenko harmonic decomposition (PHD) [17] uses the eigenvector associated with the minimum eigenvalue of the autocorrelation matrix of the target signal. The frequency is estimated as the phase of the root of the polynomial whose coefficients are corresponding to the eigenvector. The time complexity of the original PHD is $O(N^3)$ due to the calculation of eigenvectors and eigenvalues. Another PHD method in the case of a single tone is proposed by Eriksson [18], [19]. The method estimates a frequency with $O(N)$ of time complexity. But the accuracy of calculation is not higher than the other methods and this method is weak against noises.

MUSIC [20] also uses the eigenvector of the autocorrelation matrix. But the spectrum estimator is improved. The computational complexity is $O(N^3)$ due to the calculation of eigenvectors and eigenvalues.

ML estimates the frequency to minimize the probability density function (PDF) of the residue. When the number of sinusoids to be estimated is one, the method is recognized as the periodogram method. But the number of sinusoids are more than one, the maximum likelihood estimator (MLE) becomes highly nonlinear function of the unknown frequencies. It is difficult to solve the equation directly in practical methods. There are many methods [14], [15] developed to solve the problem.

IQML [22] is an approach to solve the MLE. But the computation complexity is also $O(N^3)$ due to its matrix operations.

When these methods are to be implemented with General Purpose computing on Graphics Processing Unit (GPGPU) [10], the time complexity, $O(N^3)$, is too high to be run on GPU. Because the calculation time of a kernel program run on GPU is limited. In the GHA, the sinusoidal parameters are calculated one by one and subtracted from the target signal. The approach can make the time complexity of a kernel program lower than the other approaches. It is more suitable for GPGPU than the other approaches.

GHA is used for several applications. Hirata [2] and Nakazawa [3] applied GHA to the audio coding. Hirata's method achieved high compression ratio for speech audio. Nakazawa's method achieved high quality and high speed compression using $1/12N$ octave frequency quantization for sounds of musical instruments.

Takamizawa [4] proposed a method to repair sound on a SP record. The method reduces pulse noises using Wavelet

transform as preprocessing, extracts important sound elements using GHA and resynthesizes the target signal.

A disadvantage of GHA has been its total computational cost, which limits the applications of GHA. In this paper, we propose a fast and accurate algorithm for GHA using a peak detection of Fourier spectrum, Newton's method around that peak to estimate the frequency parameters. The procedure of the frequency parameter estimation is similar to Abatzoglou's method [23] when only one sinusoid is extracted. The other parameters are estimated by minimizing residual signal power.

GHA estimates the sinusoidal parameters greedily. When a sinusoidal parameter is estimated, it will be affected by the other sinusoids. It will decrease the accuracy. To improve the accuracy of the sinusoidal parameters, a method using multi dimensional Newton's method is also proposed, which minimizes the partial difference vector of the squared error.

II. Generalized Harmonic Analysis

A. ABS method

George and Smith proposed an algorithm for GHA named Analysis By Synthesis (ABS) [5], in which sinusoids are extracted one after another as

$$x_k(n) = x_{k-1}(n) - S_k(n), \quad k = 1, 2, \cdots \tag{2}$$

where $S_k(n)$ is the k-th sinusoid given by

$$S_k(n) = A_k \sin(\omega_k n + \phi_k). \tag{3}$$

Sinusoid parameters are found by linear search to minimize the power of the residue x_k defined as

$$E_k = \sum_{n=0}^{N-1} \{x_k(n)\}^2, \tag{4}$$

where N is the frame size. Once parameters of the k-th sinusoid are estimated, that sinusoid is extracted from $x_{k-1}(n)$ as in (2). This process is repeated K times to approximate the original signal as

$$x_0(n) \quad \cong \quad x_K(n) = \sum_{k=1}^{K} A_k \sin(\omega_k n + \phi_k). \tag{5}$$

This algorithm takes much time because multiple parameters are searched simultaneously by linear search.

B. Hirata's algorithm and Muraoka's algorithm

In 1998, Hirata proposed a fast algorithm for GHA[2], in which FFT spectrum of the target signal is first used to find a rough estimate of the frequency. To estimate the frequency of the k-th sinusoid, the peak frequency of the FFT spectrum of the previous residue $x_k(n)$ is taken as the initial value and named as $f_k^{(0)}$. In the m-th search phase, the frequency is searched in the region between $f_k^{(m-1)} - F_s/(2^m N)$ and $f_k^{(m-1)} + F_s/(2^m N)$, and the frequency which minimizes the squared sum of the residue is chosen as $f_k^{(m)}$.

This method can reduce the number of search while keeping the accuracy of the estimated frequency.

In 2003, Muraoka proposed a more accurate algorithm for GHA [9] than Hirata's algorithm.

C. Noda's algorithm

We proposed a method which applies peak detection of the Fourier spectrum and binary search to improve the frequency accuracy. Our method applies peak detection of the Fourier spectrum to estimate parameters and the frequency accuracy is improved at every iteration.

1) Frequency estimation: In the first step, we estimate the frequency (angular frequency) of a sinusoid to be extracted. In this step, we use discrete time fourier transform (DTFT) of the frame defined by

$$X_k(\omega) = \sum_{n=0}^{N-1} x_k e^{-i\omega n}. \tag{6}$$

We search ω which maximize $|X_k(\omega)|$ defined by

$$|X_k(\omega)| \quad = \quad \sqrt{X_{k_r}^2(\omega) + X_{k_i}^2(\omega)} \tag{7}$$

$$X_{k_r}(\omega) \quad = \quad \sum_{n=0}^{N-1} x_k(n) \cos(\omega n) \tag{8}$$

$$X_{k_i}(\omega) \quad = \quad \sum_{n=0}^{N-1} x_k(n) \sin(\omega n), \tag{9}$$

where $X_{k_r}(\omega)$ and $X_{k_i}(\omega)$ are the real and imaginary part of (6).

Instead of DTFT, we apply FFT to a target signal to get sampled frequencies. Among FFT spectra a peak is detected and call it $\omega(0)$. The truly optimal ω is considered to be around $\omega(0)$. So the range from $\omega(0) - 2\pi/N$ to $\omega(0) + 2\pi/N$ is searched to find the optimal ω, where N is the frame size.

We assume $|X_k(\omega)|$ is convex upward in this range so that

$$\frac{\partial |X_k(\omega)|}{\partial \omega} \quad = \quad 0 \tag{10}$$

holds at the maximum ω.

The derivative is calculated as

$$\begin{aligned} \frac{\partial}{\partial \omega} |X_k(\omega)| \quad &= \quad \frac{\partial}{\partial \omega} \sqrt{X_{k_r}^2(\omega) + X_{k_i}^2(\omega)} \\ &= \quad \frac{X_{k_r}(\omega) \frac{\partial}{\partial \omega} X_{k_r}(\omega) + X_{k_i}(\omega) \frac{\partial}{\partial \omega} X_{k_i}(\omega)}{\sqrt{|X_k(\omega)|}} \\ &= \quad 0, \end{aligned} \tag{11}$$

where $\frac{\partial}{\partial \omega} X_{k_r}(\omega)$ and $\frac{\partial}{\partial \omega} X_{k_i}(\omega)$ are expressed as

$$\frac{\partial}{\partial \omega} X_{k_r}(\omega) \quad = \quad -\sum_{n=0}^{N-1} n x(n) \sin(\omega n) \tag{12}$$

$$\frac{\partial}{\partial \omega} X_{k_i}(\omega) \quad = \quad \sum_{n=0}^{N-1} n x(n) \cos(\omega n). \tag{13}$$

$\frac{\partial}{\partial \omega} |X_k(\omega)|$ is a monotonically decreasing function in this range, and ω can be found by binary search. We can narrow the search range by $1/2$ in one iteration. So calculation accuracy of ω is 2^{-M} where M is the number of iteration. In other words, we can estimate ω in time complexity $O(-N \log \varepsilon)$ where ε is tolerance of frequency.

When we search ω by binary search, the numerator of (11) is unimportant, because only the sign of (11) is considered in the binary search.

This algorithm may not work well when there are more than two local maxima of $|X_k(\omega)|$ in the range from $\omega(0)-2\pi/N$ to $\omega(0)+2\pi/N$. Otherwise it works well even if some noises, Gaussian noise for example, are mixed in the target signal. In that case, not only sinusoids which construct the target signal but also noise components are extracted.

The time complexity of this method is the same as that of Hirata's algorithm. But the number of calculation is about $1/4$ of Hirata's algorithm, because the number of sum calculation in $|X_k(\omega)|$ is half of the A_k and B_k calculations in Hirata's method, and the iteration of the binary search is half of the search routine of Hirata's method.

2) Phase and Amplitude Estimation: After estimation of the frequency, we can estimate the phase of the sinusoid simply by

$$\phi_k = \arctan(\frac{X_{k_i}(\omega_k)}{X_{k_r}(\omega_k)}). \quad (14)$$

To estimate the amplitude of the sinusoid, we use the square sum of the residue. We estimate the amplitude to minimize E_k, expressed by

$$E_k = \sum_{n=0}^{N-1}\{e_k(n)\}^2 \quad (15)$$

$$e_k(n) = x_{k-1}(n) - A_k\sin(\omega_k n+\phi_k). \quad (16)$$

To minimize it, we calculate $\frac{\partial E_k}{\partial A_k}=0$. It is derived as

$$\begin{aligned}\frac{\partial E_k}{\partial A_k} &= \frac{\partial}{\partial A_k}\sum_{n=0}^{N-1}\{x_{k-1}(n)-A_k\sin(\omega_k n+\phi_k)\}^2\\ &= \sum_{n=0}^{N-1}\{2(x_{k-1}(n)-A_k\sin(\omega_k n+\phi_k))\\ &\quad \sin(\omega_k n+\phi_k)\}\\ &= 0 \quad (17)\end{aligned}$$

$$A_k = \frac{\sum_{n=0}^{N-1}x_{k-1}(n)\sin(\omega_k n+\phi_k)}{\sum_{n=0}^{N-1}\sin^2(\omega_k n+\phi_k)}. \quad (18)$$

After calculating parameters of the sinusoid, we subtract it from the target signal. We repeat these steps until enough number of sinusoids are extracted.

3) Time Complexity: Time complexities for FFT, frequency estimation, phase estimation, amplitude estimation and subtraction are $O(N\log N)$, $O(-N\log\varepsilon)$, $O(1)$, $O(N)$ and $O(N)$, respectively. We repeat sinusoid extraction for K times. So the time complexity of the proposed method is

$$\begin{aligned}&O(K(N\log N - N\log\varepsilon+1+N+N))\\ =\ &O(NK\log\tfrac{N}{\varepsilon}).\end{aligned} \quad (19)$$

The time complexity of Hirata's algorithm is also $O(NK\log\frac{N}{\varepsilon})$. But the number of calculations is four times bigger than our proposed method.

D. Recalculation of extracted sinusoids

To speed up the calculation we used windowed DTFT whose samples are actually computed by FFT. This windowed DTFT naturally introduces an effect of the (rectangular) window, which appears as sidelobes of spectra. Those sidelobes violate the accurate estimation of sinusoid parameters.

To avoid this problem, the frequency of the extracted sinusoid is recalculated after other sinusoids are extracted. Since GHA is basically undisturbed by the window, we can recalculate the sinusoid parameters using the ones obtained in the previous section as good approximate values.

We propose two recalculation algorithms.

1) Single Recalculation: The first algorithm recalculates the parameters of the extracted sinusoids one by one.

Before a new sinusoid is extracted, the parameters of the extracted sinusoids are recalculated. The effect of the sidelobe becomes bigger in proportion to the amplitude of the sinusoid. So the parameters of the first extracted sinusoid, which has the biggest amplitude, are recalculated first to have less effect to the calculation error of the other sinusoids, and then gradually smaller sinusoids are recalculated.

The k-th extracted sinusoid is added back to the residual signal to recover $x_{k-1}(n)$. A sinusoid is extracted from $x_{k-1}(n)$ again by the method in Section II-C to replace the sinusoid parameters. This process is repeated until all the sinusoids are recalculated.

2) Double Recalculation: The second algorithm recalculates the parameters of two sinusoids adjacent in the frequency domain at the same time, to remove possible redundancy where two sinusoids have the identical frequency. This problem arises because the amplitude estimation is not optimal. That is also due to inaccuracy introduced by the windowed DTFT, which is corrected at this stage.

The algorithm is the same as the previous subsection except the following steps. Before recalculation the extracted sinusoids are sorted by their frequencies, which clarifies the existence of two identical frequencies. Possible two identical sinusoids are combined, added back to the residual signal, and the sinusoid of that frequency is recalculated just like the single recalculation algorithm. This algorithm reduces the redundancy and thus improves the overall accuracy.

3) Time Complexity: Time complexities of FFT, frequency estimation, phase estimation, amplitude estimation, addition and subtraction are $O(N\log N)$, $O(-N\log\varepsilon)$, $O(1)$, $O(N)$, $O(N)$ and $O(N)$, respectively. We repeat sinusoid extraction for $\frac{1}{2}K^2$ times. So the overall time complexity of the proposed method is

$$\begin{aligned}&O(K^2(N\log N - N\log\varepsilon+1+N+N+N))\\ =\ &O(NK^2\log\tfrac{N}{\varepsilon}).\end{aligned} \quad (20)$$

E. Parallel calculation by GPU

Graphics Processing Units (GPUs) are used for general purpose computations these days. In this case they are called General Purpose GPUs (GPGPUs) [10]. The proposed method in this paper computes a given signal in frames, and each

frame is independent from other frames. That is, the algorithm has high degree of parallelism, and it is suitable for parallel computation.

The proposed method is computed using GPU simply by allocating each frame computation to a computational unit of GPU.

1) Implementation: CUDA is an environment to support programmers to code algorithms for execution on GPU. In CUDA (compute unified device architecture) [**?**], the computation programs are divided into small programs which are called "kernel program" and kernel programs are calculated by GPU. The proposed method can be divided into five kernel programs as INITIALIZATION, FFT, EXTRACTION, ADDITION and SORT, where FFT is computed by CUFFT library included in CUDA.

In the first step, INITIALIZATION program is run to locate the target signal data to the device memory on the GPU. Then FFT, EXTRACTION, ADDITION and SORT programs are run on GPU according to the algorithm.

After the whole computations on GPU, the resultant data are copied from the GPU memory to the host memory, which are our results.

The programs except INITIALIZATION program are completed on GPU and do not transfer data between the host and GPU during the computation, and this computation-intensive nature of the proposed algorithm is quite suitable for GPU computation.

III. Frequency estimation using Newton's method

In the previous method, binary search is used to search ω. The number of iterations corresponds to needed accuracy. For example, if 10^{-15} of accuracy is needed, more than 50 iterations are needed. The number of iterations relates to the computation time.

In this section, we use Newton's method to search ω. Newton's method has quadratic convergence if its initial point is near enough to the optimal solution. Since Fourier spectrum is considered to be a good estimate of GHA, ω can be searched very fast using Newton's method.

The algorithm is another aspect of an ML-based estimation method proposed by Abatzoglou [23]. In the below description, the expression is transformed to be suitable for GHA.

A. Newton's method

Newton's method is a method to find the zero of a real-valued function. It finds the zero quickly if a near point of the zero is given.

Newton's method is represented as

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \tag{21}$$

where x_k is an approximate zero of k-th iteration, $f(x)$ is a real-valued function and $f'(x_k)$ is a first derivative of the function.

B. Frequency estimation

After applying FFT to a target signal, among FFT spectra a peak is detected and call it $\omega(0)$. Newton's method starts from the $\omega(0)$. The approximation $\omega(k+1)$ is

$$\omega(k+1) = \omega(k) - \frac{\frac{\partial |X_k(\omega_k)|}{\partial \omega_k}}{\frac{\partial^2 |X_k(\omega_k)|}{\partial \omega_k^2}}, \tag{22}$$

where

$$\begin{aligned} & \frac{\partial |X_k(\omega_k)|}{\partial \omega_k} \\ = \; & \frac{\partial}{\partial \omega_k} \sqrt{X_{k_r}^2(\omega_k) + X_{k_i}^2(\omega_k)} \\ = \; & \frac{F(\omega_k)}{G(\omega_k)} \end{aligned} \tag{23}$$

$$\begin{aligned} & \frac{\partial^2 |X_k(\omega_k)|}{\partial \omega_k^2} \\ = \; & \frac{\partial}{\partial \omega_k} \frac{F(\omega_k)}{G(\omega_k)} \\ = \; & \frac{1}{G^2(\omega_k)} \left(\frac{\partial F(\omega_k)}{\partial \omega_k} G(\omega_k) - F(\omega_k) \frac{\partial G(\omega_k)}{\partial \omega_k} \right), \end{aligned} \tag{24}$$

where

$$\begin{aligned} F(\omega_k) = \; & X_{k_r}(\omega_k) \frac{\partial}{\partial \omega_k} X_{k_r}(\omega_k) + \\ & X_{k_i}(\omega_k) \frac{\partial}{\partial \omega_k} X_{k_i}(\omega_k) \end{aligned} \tag{25}$$

$$G(\omega_k) = |X_k(\omega_k)| \tag{26}$$

$$\begin{aligned} \frac{\partial F(\omega_k)}{\partial \omega_k} = \; & X_{k_r}(\omega_k) \frac{\partial^2}{\partial \omega_k^2} X_{k_r}(\omega_k) + \left(\frac{\partial}{\partial \omega_k} X_{k_r}(\omega_k) \right)^2 + \\ & X_{k_i}(\omega_k) \frac{\partial^2}{\partial \omega_k^2} X_{k_i}(\omega_k) + \left(\frac{\partial}{\partial \omega_k} X_{k_i}(\omega_k) \right)^2 \end{aligned} \tag{27}$$

$$\frac{\partial G(\omega_k)}{\partial \omega_k} = \frac{\partial |X_k(\omega_k)|}{\partial \omega_k} \tag{28}$$

$$X_{k_r}(\omega_k) = \sum_{n=0}^{N-1} x_k(n) \cos(\omega_k n) \tag{29}$$

$$X_{k_i}(\omega_k) = \sum_{n=0}^{N-1} x_k(n) \sin(\omega_k n) \tag{30}$$

$$\frac{\partial}{\partial \omega_k} X_{k_r}(\omega_k) = - \sum_{n=0}^{N-1} n x(n) \sin(\omega_k n) \tag{31}$$

$$\frac{\partial}{\partial \omega_k} X_{k_i}(\omega_k) = \sum_{n=0}^{N-1} n x(n) \cos(\omega_k n) \tag{32}$$

$$\frac{\partial^2}{\partial \omega_k^2} X_{k_r}(\omega_k) = - \sum_{n=0}^{N-1} n^2 x(n) \cos(\omega_k n) \tag{33}$$

$$\frac{\partial^2}{\partial \omega_k^2} X_{k_i}(\omega_k) = - \sum_{n=0}^{N-1} n^2 x(n) \sin(\omega_k n). \tag{34}$$

This calculation sometimes becomes unstable. It is due to a selection of starting point and calculation error. To avoid this,

ω_k must be normalized between 0 and π after each iteration. In the normalizing, change the sign of ω_k if it is negative value, subtract 2π if it is greater than 2π and replace ω_k by $2\pi - \omega_k$ if it is greater than π.

Pseudo code to search ω is shown below. `searchOmega()` returns a value of ω searched by Newton's method, where `r` is $\omega(0)$ detected by FFT, `N` is the frame size and `x` are samples of the frame.

```
float searchOmega(float w, int N, float x[])
{
  for (int loop=0;loop<MAX_LOOP;++loop){
    float C = 0;
    float S = 0;
    float dC = 0;
    float dS = 0;
    float ddC = 0;
    float ddS = 0;
    for (int n = 0; n < N; ++n){
      floag c = cos(w * n);
      float s = sin(w * n);
      C += x[n] * c;
      S += x[n] * s;
      dC -= n * x[n] * s;
      dS += n * x[n] * c;
      ddC -= n * n * x[n] * c;
      ddS -= n * n * x[n] * s;
    }
    float f = C * dC + S * dS;
    float g = sqrt(C * C + S * S);
    float dXg = f;
    float df = C * ddC + dC * dC +
               S * ddS + dS * dS;
    float dg = f / g;
    float ddXg = (df * g - f * dg) / g;
    float dw = dX / ddX;
    w -= dw;
    if (w < 0){w *= -1;}
    while (w > PI2){w -= PI2;}
    if (w > PI){w = PI2 - w;}
  }
  return w;
}
```

1) Time Complexity: Time complexities for FFT, frequency estimation, phase estimation, amplitude estimation and subtraction are $O(N \log N)$, $O(N \log \frac{\log \varepsilon}{\log \frac{\pi}{N}})$, $O(1)$, $O(N)$ and $O(N)$, respectively, where $\frac{\pi}{N}$ is the maximum initial error of $\omega(0)$ We repeat sinusoid extraction for K times. So the time complexity of the proposed method is

$$\begin{aligned} & O(K(N \log N + N \log \frac{\log \varepsilon}{\log \frac{\pi}{N}} + 1 + N + N)) \\ = \quad & O(NK(\log N + \frac{\log \varepsilon}{\log \frac{\pi}{N}})). \end{aligned} \tag{35}$$

For example, if 10^{-15} of accuracy is needed, binary search method needs about 50 times of iterations. On the other hand, Newton's method needs

$$\frac{\log \varepsilon}{\log \frac{\pi}{N}} \quad < \quad 7 \tag{36}$$

times of iterations, where N is 512. It is much faster than binary method.

IV. Parameter adjustment using multi-dimensional Newton's method

In this section, we propose a method to improve the accuracy of the extracted parameters. In the previous method, the detected sinusoids are subtracted from the target signal one by one. When an extracted sinusoidal parameter has an error, that error affects the following parameters, and we may have larger errors in the later estimation of parameters. We propose a method to adjust the extracted sinusoidal parameters at the same time to suppress error propagation.

The purpose of GHA is to minimize the error E_k, where E_k is represented as

$$\begin{aligned} E_k \quad &= \quad \sum_{n=0}^{N-1} \{x_k(n)\}^2 \\ &= \quad \sum_{n=0}^{N-1} \{x_0(n) - \sum_{i=0}^{k-1} S_i\} \\ &= \quad \sum_{n=0}^{N-1} \{x_0(n) - \sum_{i=0}^{k-1} A_i \sin(\omega_i n + \phi_i)\} \end{aligned} \tag{37}$$

This equation represents that E_k is a function of $\{A_0, ..., A_{k-1}, \omega_0, ..., \omega_{k-1}, \phi_0, ..., \phi_{k-1}\}$. ($\{x(0), ..., x(N-1)\}$ are constant values in the equation.) This is formulated as an optimization problem. We use Newton's method to solve it

To minimize E_k, $\mathbf{p}$ must be zero, where

$$\begin{aligned} \mathbf{p} \quad = \quad & (\frac{\partial E_k}{\partial A_0}, ..., \frac{\partial E_k}{\partial A_{k-1}}, \frac{\partial E_k}{\partial \omega_0}, ..., \frac{\partial E_k}{\partial \omega_{k-1}}, \\ & \frac{\partial E_k}{\partial \phi_0}, ..., \frac{\partial E_k}{\partial \phi_{k-1}}) \end{aligned} \tag{38}$$

To approximate it, we use an approximation represented as

$$\mathbf{p}_{n+1} \quad = \quad \mathbf{p}_n - \gamma \partial E_k(\mathbf{p}_n)^{-1} E_k(\mathbf{p}_n), \tag{39}$$

where $\mathbf{p}_n$ is n-th iteration of $\mathbf{p}$ approximation, γ is a constant value which satisfies $0 < \gamma < 1$ and $\partial E_k(\mathbf{p}_n)$ is Jacobian matrix of $E_k(\mathbf{p}_n)$. $\partial E_k(\mathbf{p}_n)$ is represented as

$$\partial E_k(\mathbf{p}_n) \quad = \quad \begin{bmatrix} M_{AA} & M_{A\omega} & M_{A\phi} \\ M_{\omega A} & M_{\omega\omega} & M_{\omega\phi} \\ M_{\phi A} & M_{\phi\omega} & M_{\phi\phi} \end{bmatrix}, \tag{40}$$

where

$$M_{AA} = \begin{bmatrix} \frac{\partial^2 E_k}{\partial A_0^2} & \cdots & \frac{\partial^2 E_k}{\partial A_{k-1} \partial A_0} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 E_k}{\partial A_0 \partial A_{k-1}} & \cdots & \frac{\partial^2 E_k}{\partial A_{k-1}^2} \end{bmatrix} \tag{41}$$

$$M_{A\omega} = \begin{bmatrix} \frac{\partial^2 E_k}{\partial A_0 \partial \omega_0} & \cdots & \frac{\partial^2 E_k}{\partial A_{k-1} \partial \omega_0} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 E_k}{\partial A_0 \partial \omega_{k-1}} & \cdots & \frac{\partial^2 E_k}{\partial A_{k-1} \partial \omega_{k-1}} \end{bmatrix} \quad (42)$$

$$M_{A\phi} = \begin{bmatrix} \frac{\partial^2 E_k}{\partial A_0 \partial \phi_0} & \cdots & \frac{\partial^2 E_k}{\partial A_{k-1} \partial \phi_0} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 E_k}{\partial A_0 \partial \phi_{k-1}} & \cdots & \frac{\partial^2 E_k}{\partial A_{k-1} \partial \phi_{k-1}} \end{bmatrix} \quad (43)$$

$$M_{\omega A} = \begin{bmatrix} \frac{\partial^2 E_k}{\partial \omega_0 \partial A_0} & \cdots & \frac{\partial^2 E_k}{\partial \omega_{k-1} \partial A_0} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 E_k}{\partial \omega_0 \partial A_{k-1}} & \cdots & \frac{\partial^2 E_k}{\partial \omega_{k-1} \partial A_{k-1}} \end{bmatrix} \quad (44)$$

$$M_{\omega\omega} = \begin{bmatrix} \frac{\partial^2 E_k}{\partial \omega_0^2} & \cdots & \frac{\partial^2 E_k}{\partial \omega_{k-1} \partial \omega_0} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 E_k}{\partial \omega_0 \partial \omega_{k-1}} & \cdots & \frac{\partial^2 E_k}{\partial \omega_{k-1}^2} \end{bmatrix} \quad (45)$$

$$M_{\omega\phi} = \begin{bmatrix} \frac{\partial^2 E_k}{\partial \omega_0 \partial \phi_0} & \cdots & \frac{\partial^2 E_k}{\partial \omega_{k-1} \partial \phi_0} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 E_k}{\partial \omega_0 \partial \phi_{k-1}} & \cdots & \frac{\partial^2 E_k}{\partial \omega_{k-1} \partial \phi_{k-1}} \end{bmatrix} \quad (46)$$

$$M_{\phi A} = \begin{bmatrix} \frac{\partial^2 E_k}{\partial \phi_0 \partial A_0} & \cdots & \frac{\partial^2 E_k}{\partial \phi_{k-1} \partial A_0} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 E_k}{\partial \phi_0 \partial A_{k-1}} & \cdots & \frac{\partial^2 E_k}{\partial \phi_{k-1} \partial A_{k-1}} \end{bmatrix} \quad (47)$$

$$M_{\phi\omega} = \begin{bmatrix} \frac{\partial^2 E_k}{\partial \phi_0 \partial \omega_0} & \cdots & \frac{\partial^2 E_k}{\partial \phi_{k-1} \partial \omega_0} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 E_k}{\partial \phi_0 \partial \omega_{k-1}} & \cdots & \frac{\partial^2 E_k}{\partial \phi_{k-1} \partial \omega_{k-1}} \end{bmatrix} \quad (48)$$

$$M_{\phi\phi} = \begin{bmatrix} \frac{\partial^2 E_k}{\partial \phi_0^2} & \cdots & \frac{\partial^2 E_k}{\partial \phi_{k-1} \partial \phi_0} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 E_k}{\partial \phi_0 \partial \phi_{k-1}} & \cdots & \frac{\partial^2 E_k}{\partial \phi_{k-1}^2} \end{bmatrix}, \quad (49)$$

where

$$\frac{\partial^2 E_k}{\partial A_i^2} = 2 \sum_{n=0}^{N-1} (\frac{\partial R(n)}{\partial A_i})^2 \quad (50)$$

$$\frac{\partial^2 E_k}{\partial A_i \partial A_j} = 2 \sum_{n=0}^{N-1} \frac{\partial R(n)}{\partial A_i} \frac{\partial R(n)}{\partial A_j} \quad (51)$$

$$\frac{\partial^2 E_k}{\partial A_i \partial \omega_i} = 2 \sum_{n=0}^{N-1} (R(n) \frac{\partial^2 R(n)}{\partial A_i \partial \omega_i} + \frac{\partial R(n)}{\partial A_i} \frac{\partial R(n)}{\partial \omega_i}) \quad (52)$$

$$\frac{\partial^2 E_k}{\partial A_i \partial \omega_j} = 2 \sum_{n=0}^{N-1} \frac{\partial R(n)}{\partial A_i} \frac{\partial R(n)}{\partial \omega_j} \quad (53)$$

$$\frac{\partial^2 E_k}{\partial A_i \partial \phi_i} = 2 \sum_{n=0}^{N-1} (R(n) \frac{\partial^2 R(n)}{\partial A_i \partial \phi_i} + \frac{\partial R(n)}{\partial A_i} \frac{\partial R(n)}{\partial \phi_i}) \quad (54)$$

$$\frac{\partial^2 E_k}{\partial A_i \partial \phi_j} = 2 \sum_{n=0}^{N-1} \frac{\partial R(n)}{\partial A_i} \frac{\partial R(n)}{\partial \phi_j} \quad (55)$$

$$\frac{\partial^2 E_k}{\partial \omega_i^2} = 2 \sum_{n=0}^{N-1} \{R(n) \frac{\partial^2 R(n)}{\partial \omega_i^2} + (\frac{\partial R(n)}{\partial \omega_i})^2\} \quad (56)$$

$$\frac{\partial^2 E_k}{\partial \omega_i \partial \omega_j} = 2 \sum_{n=0}^{N-1} \frac{\partial R(n)}{\partial \omega_i} \frac{\partial R(n)}{\partial \omega_j} \quad (57)$$

$$\frac{\partial^2 E_k}{\partial \omega_i \partial \phi_i} = 2 \sum_{n=0}^{N-1} (R(n) \frac{\partial^2 R(n)}{\partial \omega_i \partial \phi_i} + \frac{\partial R(n)}{\partial \omega_i} \frac{\partial R(n)}{\partial \phi_i}) \quad (58)$$

(59)

$$\frac{\partial^2 E_k}{\partial \omega_i \partial \phi_j} = 2 \sum_{n=0}^{N-1} \frac{\partial R(n)}{\partial \omega_i} \frac{\partial R(n)}{\partial \phi_j} \quad (60)$$

$$\frac{\partial^2 E_k}{\partial \phi_i^2} = 2 \sum_{n=0}^{N-1} \{R(n) \frac{\partial^2 R(n)}{\partial \phi_i^2} + (\frac{\partial R(n)}{\partial \phi_i})^2\} \quad (61)$$

$$\frac{\partial^2 E_k}{\partial \phi_i \partial \phi_j} = 2 \sum_{n=0}^{N-1} \frac{\partial R(n)}{\partial \phi_i} \frac{\partial R(n)}{\partial \phi_j}, \quad (62)$$

where

$$R(n) = x_0(n) - \sum_{i=0}^{k-1} A_i \sin(\omega_i n + \phi_i) \quad (63)$$

$$\frac{\partial R(n)}{\partial A_i} = -\sin(\omega_i n + \phi_i) \quad (64)$$

$$\frac{\partial R(n)}{\partial \omega_i} = -A_i n \cos(\omega_i n + \phi_i) \quad (65)$$

$$\frac{\partial R(n)}{\partial \phi_i} = -A_i \cos(\omega_i n + \phi_i) \quad (66)$$

$$\frac{\partial^2 R(n)}{\partial A_i \partial \omega_i} = -n \cos(\omega_i n + \phi_i) \quad (67)$$

$$\frac{\partial^2 R(n)}{\partial A_i \partial \phi_i} = -\cos(\omega_i n + \phi_i) \quad (68)$$

$$\frac{\partial^2 R(n)}{\partial \omega_i \partial \omega_i} = A_i n^2 \sin(\omega_i n + \phi_i) \quad (69)$$

$$\frac{\partial^2 R(n)}{\partial \omega_i \partial \phi_i} = A_i n \sin(\omega_i n + \phi_i) \quad (70)$$

$$\frac{\partial^2 R(n)}{\partial \phi_i \partial \phi_i} = A_i \sin(\omega_i n + \phi_i). \quad (71)$$

Pseudo code to realize above algorithm is shown below. `adjustParameters()` improves the accuracy of the parameters using Newton's method, where `x` are samples of the frame, `N` is the frame size, `p` are the extracted sinusoidal parameters aligned like $\{A_0, ..., A_{k-1}, \omega_0, ..., \omega_{k-1}, \phi_0, ..., \phi_{k-1}\}$ and `r` is γ.

```
void adjustParameters(double x[],
 int N, double p[], double r)
{
```

```
for (int loop=0;loop<MAX_LOOP;++loop){
 double B[N];
 for (int n = 0; n < N; ++n){
  B[n] = x[n];
 }
 double BA[K][N];
 double Bw[K][N];
 double Bp[K][N];
 double BAw[K][N];
 double BAp[K][N];
 double Bww[K][N];
 double Bwp[K][N];
 double Bpp[K][N];

 for (int n = 0; n < N; ++n) {
  for (int k = 0; k < K; ++k) {
   double Ak = p[k + K * 0];
   double wk = p[k + K * 1];
   double pk = p[k + K * 2];
   double s = sin(wk * n + pk);
   double c = cos(wk * n + pk);
   B[n] -= Ak * s;
   BA[k][n] = -s;
   Bw[k][n] = -Ak * n * c;
   Bp[k][n] = -Ak * c;
   BAw[k][n] = -n * c;
   BAp[k][n] = -c;
   Bww[k][n] = Ak * n * n * s;
   Bwp[k][n] = Ak * n * s;
   Bpp[k][n] = Ak * s;
  }
 }

 double M[K3][K3];
 for (int i = 0; i < K; ++i) {
  for (int j = 0; j < K; ++j) {
   for (int n = 0; n < N; ++n) {
    if (i == j) {
     M[i+K*0][j+K*0]+=
      BA[i][n]*BA[i][n];
     M[i+K*0][j+K*1]+=
      B[n]*BAw[i][n]+BA[i][n]*Bw[i][n];
     M[i+K*0][j+K*2]+=
      B[n]*BAp[i][n]+BA[i][n]*Bp[i][n];
     M[i+K*1][j+K*1]+=
      B[n]*Bww[i][n]+Bw[i][n]*Bw[i][n];
     M[i+K*1][j+K*2]+=
      B[n]*Bwp[i][n]+Bw[i][n]*Bp[i][n];
     M[i+K*2][j+K*2]+=
      B[n]*Bpp[i][n]+Bp[i][n]*Bp[i][n];
    } else {
     M[i+K*0][j+K*0]+=BA[i][n]*BA[j][n];
     M[i+K*0][j+K*1]+=BA[i][n]*Bw[j][n];
     M[i+K*0][j+K*2]+=BA[i][n]*Bp[j][n];
     M[i+K*1][j+K*1]+=Bw[i][n]*Bw[j][n];
     M[i+K*1][j+K*2]+=Bw[i][n]*Bp[j][n];
     M[i+K*2][j+K*2]+=Bp[i][n]*Bp[j][n];
    }
   }
  }
  M[i + K * 0][j + K * 0] *= 2;
  M[i + K * 0][j + K * 1] *= 2;
  M[i + K * 0][j + K * 2] *= 2;
  M[i + K * 1][j + K * 1] *= 2;
  M[i + K * 1][j + K * 2] *= 2;
  M[i + K * 2][j + K * 2] *= 2;

  M[i+K*0][j+K*1]=M[i+K*1][j+K*0];
  M[i+K*0][j+K*2]=M[i+K*2][j+K*0];
  M[i+K*1][j+K*2]=M[i+K*2][j+K*1];
 }
}

double[] fp[K3];
for (int k = 0; k < K; ++k) {
 for (int n = 0; n < N; ++n) {
  fp[k + K * 0] += B[n] * BA[k][n];
  fp[k + K * 1] += B[n] * Bw[k][n];
  fp[k + K * 2] += B[n] * Bp[k][n];
 }
 fp[k + K * 0] *= 2;
 fp[k + K * 1] *= 2;
 fp[k + K * 2] *= 2;
}

double[] fx0 = solve(M, fp);
for (int k = 0; k < K3; ++k) {
 p[k] -= r * fx0[k];
}

for (int k = 0; k < K; ++k) {
 if (p[k] < 0) {
  p[k] *= -1;
  p[k + K * 2] += Math.PI;
 }

 if (p[k] > 1) {
  p[k+K*0]=rand()/(double)RAND_MAX;
  p[k+K*1]=rand()/(double)RAND_MAX;
  p[k+K*2]=rand()/(double)RAND_MAX;
 }
}

for (int k = 0; k < K; ++k) {
 if (p[k + K] < 0) {
  p[k + K] *= -1;
  p[k + K * 2] = PI * 2 - p[k + K * 2];
 }
 while (p[k + K] > PI * 2) {
  p[k + K] -= PI * 2;
 }
 if (p[k + K] > Math.PI) {
  p[k + K] = PI * 2 - p[k + K];
```

TABLE I
MACHINE ENVIRONMENT

CPU	Intel Core 2 Quad Q6600
MEM	2.0GB
HDD	80GB
OS	Windows Vista 64-bit
IDE	Visual Studio 2008 Eclipse 3.4.2

```
    }
   }

   for (int k = K * 2; k < K * 3; ++k) {
    while (p[k] < -PI) {
     p[k] += PI * 2;
    }
    while (p[k] > PI) {
     p[k] -= PI * 2;
    }
   }
  }
}
```

2) Time Complexity: Time complexities for creating $\partial E_k(\mathbf{p}_n)$, creating $E_k(\mathbf{p}_n)$ and solving $\partial E_k(\mathbf{p}_n)\mathbf{r} = E_k(\mathbf{p}_n)$ are $O(K^2N)$, $O(KN)$ and $O(K^3)$, respectively. So the time complexity of the proposed method is

$$\begin{aligned} & O(K^2N + KN + K^3) \\ = \quad & O(K^2N + K^3). \end{aligned} \tag{72}$$

V. EXPERIMENT

A. Experimental method

We implemented the proposed methods using CUDA[**?**] and Java, and measured the extraction accuracy and the computational time. Ten-second, 44.1kHz sampled, 16-bit, stereo, pop music was used as the target signal.

In the experiment of frequency searching, we extracted 1-128 sinusoids from each of 512-point frames of the target signal, resynthesized them and measured the PSNR between the target signal and the resynthesized signals, where the repeat count of binary search is 20 times and that of Newton's method is 5 times.

In the experiment of parameter adjustment, we extracted 1-64 sinusoids from the first frame of the target signal, applied the Newton's method and measured the PSNR between the target signal and the resynthesized signal, where γ is $1/K$ and the frame size is 512.

Microsoft Visual C++ 9.0 and NVIDIA CUDA 2.1 are used for the CUDA program and JDK 6 Update 13 is used for the Java program. Table I shows the specifications of a machine used for our experiment.

B. Results

Figure 1 - 4 show the results of the experiments.

Figure 1 shows the PSNR between the original audio signal and the signals resynthesized by the no recalculation algorithm and single recalculation algorithm. Their frequencies are searched by binary search method and Newton's method. The PSNR of the Newton's method is equal to or a bit less than binary search method. But the difference is not so much.

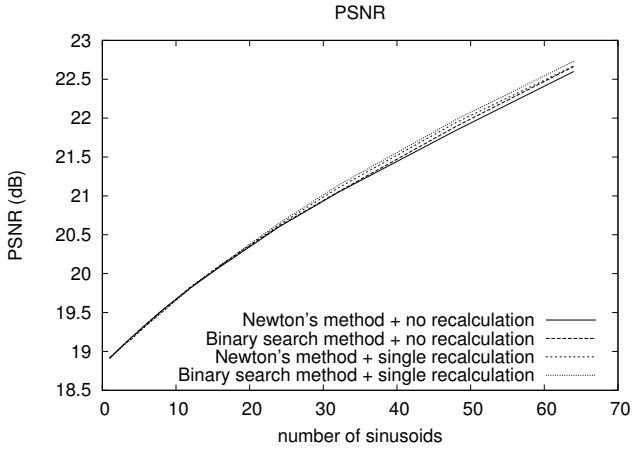

Fig. 1. PSNR of resynthesized sinusoids

Figure 2 shows the computation time of no recalculation method whose frequencies are searched by binary search method and Newton's method. The computation time of Newton's method is 1.91 times faster than binary search method. This meets the difference of the time complexities between binary search method and Newton's method.

Figure 3 shows the computation time of single recalculation method whose frequencies are searched by binary search method and Newton's method. The computation time of Newton's method is 1.72 times faster than binary search method. This also meets the difference of the time complexities between binary search method and Newton's method.

These results show that the Newton's method can esitmate frequency parameters faster than binary search. Newton's method has quadratic convergence if the starting point is enough near to the optimal solution. They mean that FFT approximates the peak of DTFT spectrum well enough.

Table II and III show the PSNR between the original audio signal and the resynthesized signal of each Newton's method iteration. The PSNR rise up to 0.68 dB in no calculation method and up to 0.48 dB in recalculation method. Almost all the PSNR rise from 0-th iteration and converge to some values. This means that Newton's method can converge the extracted sinusoidal parameters to an optimal solution. But It can not be evaluated the global optimal solution or a local optimal solution.

The PSNR of 64 sinusoids with no recalculation is disrupted. This is due to smallness of the eigenvalue of Jacobian matrix $\partial E_k(\mathbf{p}_n)$. It makes computation of $\partial E_k(\mathbf{p}_n)^{-1}$ unstable. This problem may be resolved by using another stable method to calculate $\partial E_k(\mathbf{p}_n)^{-1}$ or using quasi-Newton's method.

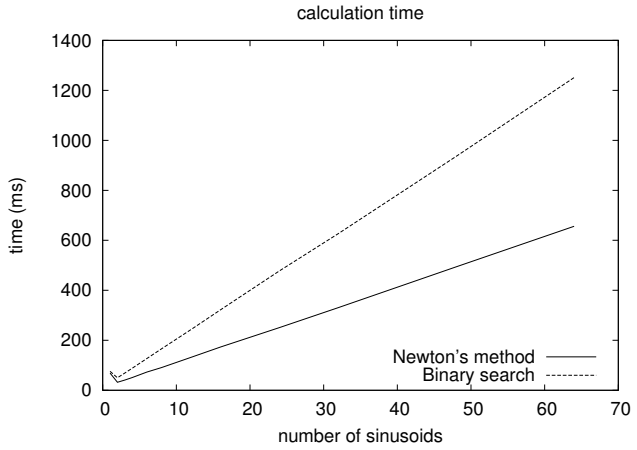

Fig. 2. Computation Time with no recalculation

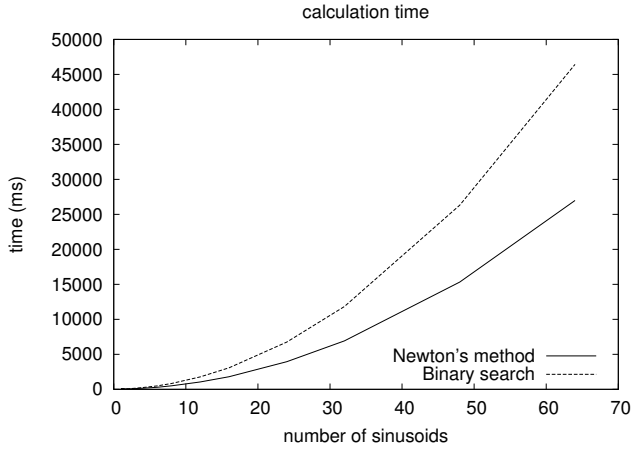

Fig. 3. Computation Time with single recalculation

VI. CONCLUSIONS

A new method for searching a frequency parameter in GHA and a parameter improving method are proposed. The method for searching frequency uses Newton's method to estimate the peak of Fourier spectrum. The parameter adjustment method uses Newton's method to optimize the extracted sinusoidal parameters.

TABLE II
PSNR OF RESYNTHESIZED SINUSOIDS WITH NO RECALCULATION METHOD

Number of sinusoids	PSNR before (dB)	PSNR after (dB)	Improvement (dB)
1	9.48	9.56	0.08
2	11.45	11.93	0.48
4	13.74	14.28	0.54
8	15.79	16.17	0.38
16	16.76	17.33	0.57
32	19.71	20.39	0.68
64	22.58	0.82	-21.76

TABLE III
PSNR OF RESYNTHESIZED SINUSOIDS WITH NO RECALCULATION METHOD

Number of sinusoids	PSNR before (dB)	PSNR after (dB)	Improvement (dB)
1	9.48	9.56	0.08
2	11.45	11.93	0.48
4	13.83	14.29	0.46
8	15.87	16.19	0.32
16	18.16	18.56	0.40
32	20.74	20.86	0.12
64	25.20	25.37	0.17

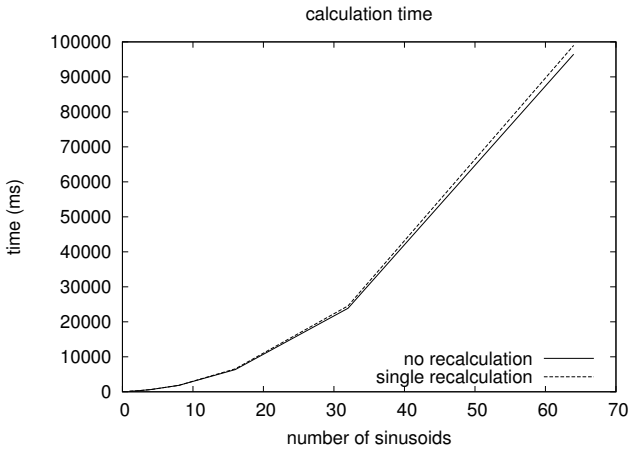

Fig. 4. Computation Time

Using the proposed methods, up to sixty four sinusoids are extracted from each frame of 10-second audio signal. Signal is resynthesized and compared with the original audio signal. The proposed frequency searching method shows 1.91 times faster than the previous method and the parameter improving method improved 0.68dB in PSNR.

REFERENCES

[1] N. Wiener, "Generalized harmonic analysis," Acta Mathematica, vol.55, pp.117–285, 1930.
[2] Y. Hirata and T. Koike, "Speech band compression using a generalized harmonic analysis," Technical report of IEICE. EA, vol.98, no.277, pp.17–24, 1998.
[3] M. Nakazawa and Y. Yamasaki, "Sound coding using 1/12n octave analysis," GITS/GITI research bulletin, vol.2002, pp.81–85, 20030731.
[4] R. Takamizawa, K. Katayama, Y. Kanda, and T. Muraoka, "Scratch noise reduction of sp record utilizing generalized harmonic analysis (gha)," IPSJ SIG Technical Reports. SLDM, vol.2004, no.102, pp.1–6, 20041021.
[5] E.B. George, "Analysis-by-synthesis/overlap-add sinusoidal modeling applied to the analysis and synthesis of musical tones," J. Audio Eng. Soc., vol.40, no.6, pp.497–515, 1992.
[6] S. Ushiyama, M. Tohyama, M. Iizuka, and Y. Hirata, "Generalized harmonic analysis of non-stationary waveforms," IEICE Technical Report. EA, vol.93, no.527, pp.39–44, 1994.
[7] M. Tohyama and T. Koike, "High resolution frequency analysis.," The Journal of the Acoustical Society of Japan, vol.54, no.8, pp.568–574, 1998.
[8] T. Terada, "Nonstationary waveform analysis and synthesis using generalized harmonic analysis," IEEE TF/TS Symp., pp.429–432, 1994.
[9] T. Muraoka and S. Kiriu, "Reduction of frequency searching processes for generalized harmonic analysis(gha)," IEICE Technical Report. DSP, vol.103, no.146, pp.1–6, 2003.

[10] gpgpu.org, "General-purpose computation on gpus (gpgpu)," http://gpgpu.org/.
[11] N. Corp. "NVIDIA CUDA Compute Unified Device Architecture Programming Guide Version 2.1," 2008.
[12] Hisayori Noda and Akinori NISHIHARA: "Fast and Accurate Generalized Harmonic Analysis and Its Parallel Computation by GPU," IEICE Trans. Fundamentals., E92-A, 3, pp.745-752, Mar. 2009.
[13] Hisayori Noda and Akinori Nishihara: "Fast Algorithm and Implementation of Generalized Harmonic Analysis," 2009 International Symposium on Multimedia and Communication Technology, JP-5, Bangkok, Jan. 2009. (Invited Paper)
[14] S. M. Kay, "Modern Spectral Estimation," Englewood Cliffs, NJ: Prentice-Hall, 1988.
[15] S. L. Marple, "Digital Spectral Analysis," Englewood Cliffs, NJ: Prentice-Hall, 1987.
[16] A. Schuster, "On the investigation of hidden periodicities with application to a supposed 26 day period of meteorological phenomena," Terrestrial Magnetism and Atmospheric Electricity, vol.3, pp.13-41, 1898.
[17] V. F. Pisarenko, "The retrieval of harmonics by linear prediction," Geophys. J. Roy. Astron. Soc., vol.33, pp.347-366, 1973.
[18] A. Eriksson, P. Stoica, "On statistical analysis of Pisarenko tone frequency estimator," Signal Process., vol.31, no.3, pp.349-353, April. 1993.
[19] K. W. Chan and H.C. So, "An exact analysis of Pisarenko's singletone frequency estimation algorithm," Signal Process., vol.83, no.3, pp.685-690, March. 2003.
[20] R. O. Schmidt, "Multiple emitter location and signal parameter estimation," IEEE Trans., vol. AP-34, pp.276-280, 1986.
[21] D. W. Tufts and R. Kumaresan, "Estimation of frequencies of multiple sinusoids: Making linear prediction perform like maximum likelihood," Proc. IEEE, vol.70, pp.675-989, Sept. 1982.
[22] V. Nagesha and S. M. Kay, "On frequency estimation with IQML algorithm," IEEE Trans. Acoust., Speech, Signal Processing, vol.42, pp.2509-2513, Sept. 1994.
[23] T. J. Abatzoglou, "A fast maximum likelihood algorithm for frequency estimation of a sinusoid based on Newton's method," IEEE Trans. Signal Process., vol.SP-33, no.1, pp.77-89, Feb. 1985.